SCIENCE CHINA
Mathematics

• ARTICLES •

March 2011 Vol. 54 No. 3: 623–632
doi: 10.1007/s11425-010-4112-6

A class of polynomially solvable 0-1 programming problems and an application

WANG Miao, XIE JinXing* & XIONG HuaChun

Department of Mathematical Sciences, Tsinghua University, Beijing 100084*, China*
*Email: jiaojian*123*@sina.com, jxie@math.tsinghua.edu.cn, xionghc*06*@gmail.com*

Received January 31, 2010; accepted June 2, 2010; published online October 18, 2010

Abstract It is well known that general 0-1 programming problems are NP-Complete and their optimal solutions cannot be found with polynomial-time algorithms unless P=NP. In this paper, we identify a specific class of 0-1 programming problems that is polynomially solvable, and propose two polynomial-time algorithms to find its optimal solutions. This class of 0-1 programming problems commits to a wide range of real-world industrial applications. We provide an instance of representative in the field of supply chain management.

Keywords 0-1 programming, polynomial-time algorithms, supply chain management

MSC(2000): 90C09

Citation: Wang M, Xie J X, Xiong H C. A class of polynomially solvable 0-1 programming problems and an application. Sci China Math, 2011, 54(3): 623–632, doi: 10.1007/s11425-010-4112-6

1 Introduction

For the large scale 0-1 programming problems, people usually suffer from the so-called "dimension disaster", i.e., the time used for finding the exact solution increases exponentially with respect to the scale of the problem (see, e.g., [4] for details). When this happens, various heuristic approaches, such as branch and bound (e.g., [7]), genetic algorithms (e.g., [3]), and probabilistic search (e.g., [2]) should be used to find the approximate solutions instead of the exact one. For the 0-1 programming problems heuristic algorithms are not always needed, especially with some instances of special structures. Padberg [5] showed that some special set packing problems, which can be modeled as 0-1 programming problems with perfect 0-1 coefficient matrix, can be solved by solving their linear programming (LP) relaxation problems with a polynomial-time algorithm. Similar results hold for the set covering and the set partitioning problems with ideal matrices (see [6] for details). Bilitzky and Sadeh [1] identified sufficient conditions on the coefficient matrix under which the linear 0-1 programming problem can be polynomially solved by solving its LP relaxation problem.

The papers [1, 5, 6] concerned about establishing efficient algorithms to find the optimal solutions for the special linear 0-1 programming problems. While in this paper, we identify a class of nonlinear 0-1 programming problems that can be solved by polynomial-time algorithms. Specifically, we consider the following 0-1 programming problem:

$$\text{Min} \quad f(x) = \sum_{j=1}^{2} \left(b_j \sum_{i=1}^{n} a_i x_{ji} \right)^s - \sum_{j=1}^{2} \sum_{i=1}^{n} c_{ji} x_{ji}, \tag{1}$$

*Corresponding author

$$\text{s.t.} \quad x_{2i} \geqslant x_{1i}, \quad \text{for any } i = 1, 2, \ldots, n,$$
$$x = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n}) \in \{0, 1\}^{2n},$$

where $0 < s < 1$, $a_i > 0$, $b_j > 0$, $c_{ji} > 0$, and n is an integer. In the following discussion, we will provide two polynomial-time algorithms to solve this problem and present one of its applications in supply chain management.

The rest of this paper proceeds as follows. In Section 2, we propose and analyze two polynomial-time algorithms to solve Problem (1). In Section 3, we give a real-world industrial application of Problem (1). In Section 4, we extend the results in Section 2 to a more general case.

2 Algorithms

For convenience, we assume that there are n items, which are indexed by $1, 2, \ldots, n$. Each item i possesses attributes a_i and c_{ji} $(j = 1, 2)$. Denote the set consisting of all items by $S = \{1, 2, \ldots, n\}$. For any feasible solution $x = (x_{11}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n})$ of Problem (1), define $S_1(x) = \{i \mid x_{1i} = x_{2i} = 1\}$, $S_2(x) = \{i \mid x_{1i} = 0, x_{2i} = 1\}$, $S_3(x) = \{i \mid x_{1i} = x_{2i} = 0\}$. Clearly, $S_1(x) \cup S_2(x) \cup S_3(x) = S$ and $S_k(x) \cap S_l(x) = \emptyset$ for any $k \neq l$. Sort all items in the ascending order by c_{1i}/a_i, and define $R_1(i)$ as the ranking position of item i $(i = 1, 2, \ldots, n)$. Sort all items in the ascending order by c_{2i}/a_i, and define $R_2(i)$ as the ranking position of item i $(i = 1, 2, \ldots, n)$. Sort all items in the ascending order by $(c_{1i} + c_{2i})/a_i$, and define $R_3(i)$ as the ranking position of item i $(i = 1, 2, \ldots, n)$. For example, if $k_1 = \arg\max_i\{c_{1i}/a_i \mid i = 1, 2, \ldots, n\}$, then $R_1(k_1) = n$.

Theorem 2.1. *Assume $R_1(i)$, $R_2(i)$ and $R_3(i)$, $i = 1, 2, \ldots, n$, are determined. Then any optimal solution x^* for Problem* (1) *satisfies the following properties*:

(a) $R_1(k) < R_1(l)$ *for any* $k \in S_2(x^*), l \in S_1(x^*)$;

(b) $R_2(k) < R_2(l)$ *for any* $k \in S_3(x^*), l \in S_2(x^*)$;

(c) $R_3(k) < R_3(l)$ *for any* $k \in S_3(x^*), l \in S_1(x^*)$.

Proof. We prove it by contradiction. Suppose there exists an optimal solution x^* violating at least one of the three properties, e.g., (c) (Here we only provide the proof when x^* violates (c); proofs for the others with (a) or (b) violated are similar).

Since x^* violates (c), there exist $k \in S_3(x^*)$ and $l \in S_1(x^*)$ such that $(c_{1k} + c_{2k})/a_k \geqslant (c_{1l} + c_{2l})/a_l$. By the definitions of $S_3(x^*)$ and $S_1(x^*)$, we have $x^*_{1k} = x^*_{2k} = 0$, $x^*_{1l} = x^*_{2l} = 1$. Choose $z = (z_{11}, \ldots, z_{1n}, z_{21}, \ldots, z_{2n}) \in \{0, 1\}^{2n}$, where $z_{1k} = z_{2k} = 1$ and $z_{ji} = x^*_{ji}$ for any $j = 1, 2$ and $i \neq k$. The theorem is clearly true by contradiction if it holds that $f(z) < f(x^*)$, which is equivalent to

$$\begin{aligned}
&\left(b_1 \sum_{i\neq k,l} a_i x^*_{1i} + b_1 a_l + b_1 a_k\right)^s - \sum_{i\neq k,l} c_{1i} x^*_{1i} - c_{1l} - c_{1k} \\
&\quad + \left(b_2 \sum_{i\neq k,l} a_i x^*_{2i} + b_2 a_l + b_2 a_k\right)^s - \sum_{i\neq k,l} c_{2i} x^*_{2i} - c_{2l} - c_{2k} \\
&< \left(b_1 \sum_{i\neq k,l} a_i x^*_{1i} + b_1 a_l\right)^s - \sum_{i\neq k,l} c_{1i} x^*_{1i} - c_{1l} + \left(b_2 \sum_{i\neq k,l} a_i x^*_{2i} + b_2 a_l\right)^s - \sum_{i\neq k,l} c_{2i} x^*_{2i} - c_{2l}. \qquad (2)
\end{aligned}$$

Denote $A_1 = b_1 \sum_{i\neq k,l} a_i x^*_{1i}$, $A_2 = b_2 \sum_{i\neq k,l} a_i x^*_{2i}$. Then Inequality (2) is equivalent to

$$(A_1 + b_1 a_l + b_1 a_k)^s - (A_1 + b_1 a_l)^s + (A_2 + b_2 a_l + b_2 a_k)^s - (A_2 + b_2 a_l)^s < c_{1k} + c_{2k}. \qquad (3)$$

Consider the function $g(\eta) = \eta^s$. By the Mean Value Theorem, we know that there exist $\beta_1 \in (A_1 + b_1 a_l, A_1 + b_1 a_l + b_1 a_k)$, $\beta_2 \in (A_2 + b_2 a_l, A_2 + b_2 a_l + b_2 a_k)$ such that

$$(A_1 + b_1 a_l + b_1 a_k)^s - (A_1 + b_1 a_l)^s = b_1 a_k \cdot s\beta_1^{s-1}$$

and

$$(A_2 + b_2 a_l + b_2 a_k)^s - (A_2 + b_2 a_l)^s = b_2 a_k \cdot s\beta_2^{s-1}.$$

Thus, Inequality (3) is equivalent to

$$b_1 s\beta_1^{s-1} + b_2 s\beta_2^{s-1} < (c_{1k} + c_{2k})/a_k. \tag{4}$$

To prove this theorem, we only need to prove Inequality (4).

Choose $y = (y_{11}, \ldots, y_{1n}, y_{21}, \ldots, y_{2n}) \in \{0,1\}^{2n}$, where $y_{1l} = y_{2l} = 0$ and $y_{ji} = x^*_{ji}$ for any $j = 1, 2$ and $i \neq l$. Since x^* is the optimal solution of Problem (1), we have $f(x^*) \leqslant f(y)$. Thus, we have

$$(A_1 + b_1 a_l)^s - A_1^s + (A_2 + b_2 a_l)^s - A_2^s \leqslant c_{1l} + c_{2l}. \tag{5}$$

By the Mean Value Theorem, there exist $\alpha_1 \in (A_1, A_1 + b_1 a_l)$ and $\alpha_2 \in (A_2, A_2 + b_2 a_l)$ such that $(A_1 + a_l b_1)^s - A_1^s = a_l b_1 \cdot s\alpha_1^{s-1}$ and $(A_2 + a_l b_2)^s - A_2^s = a_l b_2 \cdot s\alpha_2^{s-1}$. Thus, Inequality (5) is equivalent to

$$b_1 \cdot s\alpha_1^{s-1} + b_2 \cdot s\alpha_2^{s-1} \leqslant (c_{1l} + c_{2l})/a_l. \tag{6}$$

For $a_i > 0$ $(i = 1, 2, \ldots, n)$ and $b_j > 0$ $(j = 1, 2)$, we have $\beta_1 > \alpha_1$, $\beta_2 > \alpha_2$. Note that $0 < s < 1$. Then $s\eta^{s-1}$ is strictly decreasing in η, which together with the condition that $(c_{1l} + c_{2l})/a_l \leqslant (c_{1k} + c_{2k})/a_k$ implies

$$b_1 \cdot s\beta_1^{s-1} + b_2 \cdot s\beta_2^{s-1} < b_1 \cdot s\alpha_1^{s-1} + b_2 \cdot s\alpha_2^{s-1} \leqslant (c_{1l} + c_{2l})/a_l \leqslant (c_{1k} + c_{2k})/a_k.$$

Therefore, Inequality (4) is true. This completes the proof. □

Theorem 2.1 characterizes the necessary conditions for the optimal solution of Problem (1).

Corollary 2.1. *Assume that $R_1(i)$, $R_2(i)$ and $R_3(i), i = 1, 2, \ldots, n$, are determined and that x satisfies the three properties in Theorem* 2.1.

(a) *Let $k = \max\{R_3(i) \mid i \in S_3(x)\}$ and define $G = \{i \in S \mid R_3(i) \leqslant k\}$. Then we have $S_3(x) \subseteq G$ and $S_1(x) \cap G = \emptyset$;*

(b) *Let $p = \max\{R_2(i) \mid i \in S_3(x)\}$ and define $P_3 = \{i \in G \mid R_2(i) \leqslant p\}$. Then we have $S_3(x) = P_3$;*

(c) *Let $q = \max\{R_1(i) \mid i \in S_2(x)\}$ and define $P_2 = \{i \in S\backslash P_3 \mid R_2(i) \leqslant q\}$. Then we have $S_2(x) = P_2$.*

Proof. Part (a). If $S_3(x) \nsubseteq G$, then there exists an item i in $S_3(x)$ such that $R_3(i) > k$, which contradicts the definition of k. If $S_1(x) \cap G \neq \emptyset$, then there exists an item i in $S_1(x)$ such that $R_3(i) \leqslant k$. Clearly, $R_3(i) \neq k$, since the item i_0 that satisfies $R_3(i_0) = k$ belongs to $S_3(x)$ and $S_1(x) \cap S_3(x) = \emptyset$. Therefore, we have $R_3(i) < R_3(i_0)$ with $i \in S_1(x)$ and $i_0 \in S_3(x)$, which contradicts with Theorem 2.1.

Part (b). Using a similar method to that in Part (a), one can prove that $S_3(x) \subseteq P_3$ and $S_2(x) \cap P_3 = \emptyset$. Note that $P_3 \subseteq G$ and $S_1(x) \cap G = \emptyset$, then we have $P_3 \cap (S_2(x) \cup S_1(x)) = \emptyset$, which, together with the facts $S_1(x) \cup S_2(x) \cup S_3(x) = S$ and $S_k(x) \cap S_l(x) = \emptyset$ for $k \neq l$, indicates $P_3 \subseteq S_3(x)$. This completes the proof of Part (b).

The proof of Part (c) is similar to that of Part (b), so we omit it here. □

Based on Corollary 2.1, we propose the following algorithm.

Algorithm 2.1.

Step 1: Solve $R_1(i)$, $R_2(i)$ and $R_3(i)$ $(i = 1, 2, \ldots, n)$ by some sorting procedure.

Let $k = 0$, $x^* = (0, 0, \ldots, 0)$ and $v^* = 0$.

Step 2: Let

$$G = \begin{cases} \emptyset, & \text{if } k = 0; \\ G \cup \{i \mid R_3(i) = k\}, & \text{if } k > 0. \end{cases}$$

For $p = 0, 1, \ldots, k$

Let $P_3 = \{i_1, \ldots, i_p\} \subseteq G$, where $i_1, \ldots, i_p$ are p items such that $R_2(i_1), \ldots, R_2(i_p)$ are the p smallest numbers in $\{R_2(i) \mid i \in G\}$.

For $q = 0, 1, \ldots, n - p$

$P_2 = \{i'_1, \ldots, i'_q\} \subseteq S\backslash P_3$, where $i'_1, \ldots, i'_q$ are q items such that $R_1(i'_1), \ldots, R_1(i'_q)$ are the q smallest numbers in $\{R_1(i) \mid i \in S\backslash P_3\}$.

Let $P_1 = S\backslash(P_2 \cup P_3)$.
Define $x = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n})$ such that $x_{1i} = x_{2i} = 1$ if $i \in P_1$; $x_{1i} = 0, x_{2i} = 1$ if $i \in P_2$; $x_{1i} = x_{2i} = 0$ if $i \in P_3$.
Calculate $v = f(x)$.
If $v < v^*$, then let $x^* = x$, and $v^* = v$.

End

End

Step 3: Let $k = k + 1$.
If $k \leqslant n$, go to Step 2.
Else, stop and output x^* as an optimal solution and v^* as the optimal value.

Theorem 2.2. *Algorithm* 2.1 *is able to find the optimal solution of Problem* (1). *The complexity of Algorithm* 2.1 *is* $O(n^3)$.

Proof. To show that Algorithm 2.1 can find the optimal solution of Problem (1), we only need to prove that all feasible solutions satisfying properties (a)–(c) in Theorem 2.1 (for $R_1(i),\ R_2(i),\ R_3(i)$ generated in Step 1 of Algorithm 2.1) are examined by Algorithm 2.1.

For any feasible solution x satisfying properties (a)–(c) in Theorem 2.1, let $k_0 = \max\{R_3(i) \mid i \in S_3(x)\}$, and $p_0,\ q_0$ be the total numbers of elements in $S_3(x)$ and $S_2(x)$, respectively. Then x is examined by Algorithm 2.1 when $k = k_0$, $p = p_0$ and $q = q_0$ (in this case, $P_1 = S_1(x),\ P_2 = S_2(x),\ P_3 = S_3(x)$).

Clearly, Step 1 runs in $O(n \log n)$, and Step 2 runs in $O(n^2)$ for any given k. Note that k varies from 0 to n, then Algorithm 2.1 possesses a complexity of $O(n^3) = O(\max\{n \log n, n^3\})$. □

Next, we will propose another algorithm with a lower complexity. Before presenting this algorithm, we will provide two sub-algorithms, which are called in this algorithm.

Sub-Algorithm 2.1.
Input: A set $G \subseteq S$.
Step 1: Denote the total number of elements in G by $|G|$. Sort the items in G in ascending order by c_{1i}/a_i, and save the ranking position of each item i in G as $R(i)$.
Let $k = 1$, $F_1^* = \emptyset$, $F_2^* = G$, $v^* = (b_1 \sum_{i \in G} a_i)^s - \sum_{i \in G} c_{1i}$.
Step 2: Let $F_1 = \{i_1, \ldots, i_k\}$ such that $R(i_1) = 1, \ldots, R(i_k) = k$, and $F_2 = G\backslash F_1$.
Calculate $v = (b_1 \sum_{i \in F_2} a_i)^s - \sum_{i \in F_2} c_{1i}$.
If $v < v^*$, then let $v^* = v$, $F_1^* = F_1$, $F_2^* = F_2$.
Step 3: Let $k = k + 1$.
If $k \leqslant |G|$, go to Step 2.
Else, stop and output F_1^*, F_2^*, v^*.

Sub-Algorithm 2.2.
Input: A set $G \subseteq S$.
Step 1: Denote the total number of elements in G by $|G|$. Sort the items in G in ascending order by c_{2i}/a_i, and save the ranking position of each item i in G as $R(i)$.
Let $k = 1$, $F_1^* = \emptyset$, $F_2^* = G$, $v^* = (b_2 \sum_{i \in S} a_i)^s - \sum_{i \in S} c_{2i}$.
Step 2: Let $F_1 = \{i_1, \ldots, i_k\}$ such that $R(i_1) = 1, \ldots, R(i_k) = k$, and $F_2 = G\backslash F_1$.
Calculate $v = (b_2 \sum_{i \in F_2} a_i + b_2 \sum_{i \in S\backslash G} a_i)^s - \sum_{i \in F_2} c_{2i} - \sum_{i \in S\backslash G} c_{2i}$.
If $v < v^*$, then let $v^* = v$, $F_1^* = F_1$, $F_2^* = F_2$.
Step 3: Let $k = k + 1$.
If $k \leqslant |G|$, go to Step 2.
Else, stop and output F_1^*, F_2^*, v^*.

Based on these two sub-algorithms, we present the main algorithm as follows.

Algorithm 2.2.
Step 1: Solve $R_1(i)$, $R_2(i)$, and $R_3(i)$ $(i = 1, 2, \ldots, n)$ by some sorting procedure.
Let $k = 0$, $x^* = (0, 0, \ldots, 0)$ and $v^* = 0$.

Step 2: Let

$$G_1 = \begin{cases} \emptyset, & \text{if } k = 0; \\ G_1 \cup \{i \mid R_3(i) = k\}, & \text{if } k > 0, \end{cases} \quad G_2 = \begin{cases} S, & \text{if } k = 0; \\ G_2 \backslash \{i \mid R_3(i) = k\}, & \text{if } k > 0. \end{cases}$$

Call Sub-Algorithm 2.1 with input G_2, and save the output $F_1^* \to \tilde{P}_2$, $F_2^* \to P_1$, $v^* \to v_2$.
Call Sub-Algorithm 2.2 with input G_1, and save the output $F_1^* \to P_3$, $F_2^* \to \bar{P}_2$, $v^* \to v_1$.
Let $P_2 = \bar{P}_2 \cup \tilde{P}_2$.
Define $x = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{2n})$ such that $x_{1i} = x_{2i} = 1$ if $i \in P_1$; $x_{1i} = 0$, $x_{2i} = 1$ if $i \in P_2$; $x_{1i} = x_{2i} = 0$ if $i \in P_3$.
If $v_1 + v_2 < v^*$, then let $x^* = x$ and $v^* = v_1 + v_2$.

Step 3: Let $k = k + 1$.
If $k \leqslant n$, go to Step 2.
Else stop and output x^* as an optimal solution and v^* as the optimal value.

Here we show that Algorithm 2.2 indeed finds the optimal solution of Problem (1).

Lemma 2.1. *Assume that $R_1(i)$, $R_2(i)$ and $R_3(i)$, $i = 1, 2, \ldots, n$, are determined. For any optimal solution x^* of Problem* (1)*, denote $k^* = \max\{R_3(i) \mid i \in S_3(x^*)\}$. Then we have*
(1) $x_{1i}^* = 0$, *for all* $i \in \{i \mid R_3(i) \leqslant k^*\}$;
(2) $x_{2i}^* = 1$, *for all* $i \in \{i \mid R_3(i) > k^*\}$.

Proof. It is obvious from Theorem 2.1 and the definitions of x^*, k^* and $S_3(x^*)$. □

Lemma 2.2. *For any given input $G \subseteq S$, let*

$$g_1(F) = \left(b_1 \sum_{i \in F} a_i\right)^s - \sum_{i \in F} c_{1i}, \quad g_2(F) = \left(b_2 \sum_{i \in F} a_i + b_2 \sum_{i \in S \backslash G} a_i\right)^s - \sum_{i \in F} c_{2i}$$

be two functions defined on $\{F \mid F \subseteq G\}$.
(a) *The output F_2^* of Sub-Algorithm* 2.1 *satisfies $g_1(F_2^*) = \min\{g_1(F) \mid F \subseteq G\}$;*
(b) *The output F_2^* of Sub-Algorithm* 2.2 *satisfies $g_2(F_2^*) = \min\{g_2(F) \mid F \subseteq G\}$.*

Proof. The proof is similar to that of Theorem 2.1. □

Theorem 2.3. *Algorithm* 2.2 *is able to find the optimal solution of Problem* (1)*. The complexity of Algorithm* 2.2 *is $O(n^2)$.*

Proof. We only need to prove that any solution that satisfies the two properties in Lemma 2.1 can be dominated by some solution examined in Algorithm 2.2.

For any feasible solution y satisfying the two properties in Lemma 2.1, let $k_0 = \max\{R_3(i) \mid i \in S_3(y)\}$, $\bar{S}_2(y) = \{i \in S_2(y) \mid R_3(i) < k_0\}$, $\tilde{S}_2(y) = S_2(y) \backslash \bar{S}_2(y)$. Consider Step $k = k_0$ of Algorithm 2.2. Denote by $x^{(k_0)}$ the solution generated in Step $k = k_0$ of Algorithm 2.2. Then $G_1^{(k_0)}$, $G_2^{(k_0)}$, $P_1^{(k_0)}$, $\tilde{P}_2^{(k_0)}$, $\bar{P}_2^{(k_0)}$, $P_3^{(k_0)}$ denote the corresponding items G_1, G_2, P_1, $\tilde{P}_2$, $\bar{P}_2$, P_3 generated in Step $k = k_0$ of Algorithm 2.2. To prove that Algorithm 2.2 can find an optimal solution of Problem (1), we only need to prove that $f(x^{(k_0)}) \leqslant f(y)$.

By the definition of y, we have $y_{1i} = 0$ for all $i \in \{i \mid R_3(i) \leqslant k_0\} = G_1^{(k_0)}$, which means $i \notin S_1(y)$ for any $i \in \{i \mid R_3(i) \leqslant k_0\} = G_1^{(k_0)}$. Similarly, we have $i \notin S_3(y)$ for any $i \in \{i \mid R_3(i) > k_0\} = G_2^{(k_0)}$. Thus, it holds that $S_3(y) \cup \bar{S}_2(y) = G_1^{(k_0)}$ and $S_1(y) \cup \tilde{S}_2(y) = G_2^{(k_0)}$. Since y satisfies the properties (1)–(2) of Lemma 2.1, we obtain the following equation:

$$\begin{aligned} f(y) &= \left(b_1 \sum_{i \in G_1^{(k_0)}} a_i y_{1i} + b_1 \sum_{i \in G_2^{(k_0)}} a_i y_{1i}\right)^s + \left(b_2 \sum_{i \in G_1^{(k_0)}} a_i y_{2i} + b_2 \sum_{i \in G_2^{(k_0)}} a_i y_{2i}\right)^s \\ &\quad - \sum_{i \in G_1^{(k_0)}} c_{1i} y_{1i} - \sum_{i \in G_2^{(k_0)}} c_{1i} y_{1i} - \sum_{i \in G_1^{(k_0)}} c_{2i} y_{2i} - \sum_{i \in G_2^{(k_0)}} c_{2i} y_{2i} \\ &= \left(b_1 \sum_{i \in G_2^{(k_0)}} a_i y_{1i}\right)^s + \left(b_2 \sum_{i \in G_1^{(k_0)}} a_i y_{2i} + A\right)^s - \sum_{i \in G_2^{(k_0)}} c_{1i} y_{1i} - \sum_{i \in G_1^{(k_0)}} c_{2i} y_{2i} - B \end{aligned}$$

$$= \left(b_1 \sum_{i\in S_1(y)} a_i\right)^s + \left(b_2 \sum_{i\in \bar{S}_2(y)} a_i + A\right)^s - \sum_{i\in S_1(y)} c_{1i} - \sum_{i\in \bar{S}_2(y)} c_{2i} - B,$$

where $A = b_2 \sum_{i\in G_2^{(}k_0)} a_i$, $B = \sum_{i\in G_2^{(}k_0)} c_{2i}$. Noticing that $x_{1i}^{(k_0)} = 0$ for $i \in G_1^{(k_0)}$ and $x_{2i}^{(k_0)} = 1$ for $i \in G_2^{(k_0)}$ (see Algorithm 2.2), we then have the following inequality by Lemma 2.2:

$$\begin{aligned} f(y) &= \left(b_1 \sum_{i\in S_1(y)} a_i\right)^s + \left(b_2 \sum_{i\in \bar{S}_2(y)} a_i + A\right)^s - \sum_{i\in S_1(y)} c_{1i} - \sum_{i\in \bar{S}_2(y)} c_{2i} - B \\ &\geqslant \left(b_1 \sum_{i\in P_1^{(k_0)}} a_i\right)^s + \left(b_2 \sum_{i\in \bar{P}_2^{(k_0)}} a_i + A\right)^s - \sum_{i\in P_1^{(k_0)}} c_{1i} - \sum_{i\in \bar{P}_2^{(k_0)}} c_{2i} - B \\ &= \left(b_1 \sum_{i\in G_2^{(k_0)}} a_i x_{1i}^{(k_0)}\right)^s + \left(b_2 \sum_{i\in G_1^{(k_0)}} a_i x_{2i}^{(k_0)} + A\right)^s - \sum_{i\in G_2^{(k_0)}} c_{1i} x_{1i}^{(k_0)} - \sum_{i\in G_1^{(k_0)}} c_{2i} x_{2i}^{(k_0)} - B \\ &= f(x^{k_0}), \end{aligned}$$

which indicates that y is dominated by $x^{(k_0)}$. This implies that Algorithm 2.2 can find an optimal solution of Problem (1).

Clearly, Step 1 runs in $O(n \log n)$, while Step 2 runs in $O(n)$ for a given k, in which Sub-Algorithm 2.1 requires $n-k+1$ times of calculations, and Sub-Algorithm 2.2 requires $k+1$ times of calculations. Note that k varies from 0 to n, then Algorithm 2.2 possesses with a complexity of $O(n^2) = O(\max\{n \log n, n^2\})$. The proof is complete. □

Theorems 2.2 and 2.3 suggest that the "dimension disaster" will not be a problem, because the specific class of 0-1 programming problems described by Problem (1) can be solved by the two polynomial-time algorithms proposed above. Consequently, whenever a large-scale problem in industrial practice can be modeled as a special case of Problem (1), the optimal solution can be obtained with a polynomial-time algorithm.

3 An application

In this section, we provide a real-world industrial example. Early order commitment (EOC) is one of the coordinating strategies in supply chain management. Zhao et al. [10] conducted, by simulations, a comprehensive study of the impacts of EOC on supply chain performance under various operational conditions. The first analytical model on EOC was developed in [11], which studied the effectiveness of EOC in a two-level supply chain consisting of a single manufacturer and a single retailer. Xie et al. [8] extended the results of [11] to a two-level supply chain with a single supplier and multiple retailers. However, in [8], the authors did not provide any polynomial-time algorithm to find the optimal EOC periods to minimize the expected holding and shortage cost per period for the whole supply chain. Xiong et al. [9] provided a polynomial-time algorithm to find such optimal EOC periods by polynomially solving a class of 0-1 programming problems, which is a special case of Problem (1) in the present paper.

All the above papers discussed two-level supply chains, while in this section, we consider a three-level supply chain, consisting of a supplier, a wholesaler and multiple retailers. Denote L_s as the manufacturing lead time of the supplier, L_w as the delivery lead time from the supplier to the wholesaler, and L_i as the delivery lead time from the wholesaler to the retailer i. Under the EOC strategy, retailer i $(i = 1, 2, \ldots, n)$ places its order x_i periods in advance, $0 \leqslant x_i \leqslant L_s + L_w + 2$. When $L_s + 1 < x_i \leqslant L_s + L_w + 2$, the wholesaler will share the EOC information of retailer i to the supplier. Denote $x = (x_1, x_2, \ldots, x_n)$. Following the similar method to that in [11], we have that the total cost of the three-level supply chain (for i.i.d. demand over time) is

$$SC_1(x) = r_s \sqrt{\sum_{i=1}^{n} \sigma_i^2 (L_s + 1 - (x_i - L_w - 1)^+)} + r_w \sqrt{\sum_{i=1}^{n} \sigma_i^2 (L_w + 1 - x_i)^+} + \sum_{i=1}^{n} r_i \sigma_i \sqrt{L_i + x_i + 1}, \quad (7)$$

where r_s is the supplier's cost parameter, r_w is the wholesaler's cost parameter, r_i $(i = 1, 2, \ldots, n)$ is the cost parameter of retailer i, and σ_i^2 $(i = 1, 2, \ldots, n)$ is the variance of the demand faced by retailer i. The following proposition characterizes the structure of the retailers' optimal EOC strategies for the supply chain.

Proposition 3.1. *In the three-level supply chain, the optimal* EOC *periods* x_k $(k = 1, 2, \ldots, n)$ *minimizing* $SC_1(x)$ *defined in Equation* (7) *must be in the set* $\{0, L_w + 1, L_s + L_w + 2\}$.

Proof. When $0 \leqslant x_k \leqslant L_w + 1$ for some $k \in \{1, 2, \ldots, n\}$, the second partial derivative of $SC_1(x)$ defined in Equation (7) with respect to x_k is

$$\frac{\partial^2 SC_1(x)}{\partial x_k^2} = -\frac{r_w \sigma_k^4}{4(\sum_{i \neq k} \sigma_i^2 (L_w + 1 - x_i)^+ + L_w + 1 - x_k)^{3/2}} - \frac{r_k \sigma_k}{4(L_k + x_k + 1)^{3/2}} < 0, \tag{8}$$

which indicates that the $SC_1(x)$ is concave in x_k. Thus, to minimize $SC_1(x)$ in Equation (7), x_k must be either 0 or $L_w + 1$.

When $L_w + 1 \leqslant x_k \leqslant L_s + L_w + 2$ for some $k \in \{1, 2, \ldots, n\}$, it can be proved similarly that $SC_1(x)$ is also concave with respect to x_k. Therefore, to minimize $SC_1(x)$, one only needs to choose x_k in $\{L_w + 1, L_s + L_w + 2\}$.

From the above, we have that x_k must be chosen from the set $\{0, L_w + 1, L_s + L_w + 2\}$, for $k = 1, 2, \ldots, n$. □

Define $y = (y_{11}, y_{12}, \ldots, y_{1n}, y_{21}, \ldots, y_{2n}, y_{31}, \ldots, y_{3n})$ such that

$$y_{1i} = \begin{cases} 1, & x_i = 0, \\ 0, & \text{others}, \end{cases} \quad y_{2i} = \begin{cases} 1, & x_i = L_w + 1, \\ 0, & \text{others}, \end{cases} \quad y_{3i} = \begin{cases} 1, & x_i = L_s + L_w + 2, \\ 0, & \text{others}. \end{cases}$$

Since each x_i cannot be two values at the same time, $y_{1i} + y_{2i} + y_{3i} = 1$ $(i = 1, 2, \ldots, n)$, Equation (7) can be expressed as:

$$\begin{aligned} SC_2(y) &= SC_1(x) \\ &= r_s \sqrt{\sum_{i=1}^n \sigma_i^2 (L_s + 1)(1 - y_{3i})} + r_w \sqrt{\sum_{i=1}^n \sigma_i^2 (L_w + 1) y_{1i}} \\ &\quad + \sum_{i=1}^n r_i \sigma_i y_{1i} \sqrt{L_i + 1} + \sum_{i=1}^n r_i \sigma_i y_{2i} \sqrt{L_i + L_w + 2} + \sum_{i=1}^n r_i \sigma_i y_{3i} \sqrt{L_s + L_w + L_i + 3}. \end{aligned} \tag{9}$$

Replacing y_{3i} by $1 - y_{1i} - y_{2i}$, we simplify Equation (9) to

$$\begin{aligned} SC_2(y) &= r_s \sqrt{\sum_{i=1}^n \sigma_i^2 (L_s + 1)(y_{1i} + y_{2i})} + r_w \sqrt{\sum_{i=1}^n \sigma_i^2 (L_w + 1) y_{1i}} \\ &\quad + \sum_{i=1}^n r_i \sigma_i \sqrt{L_s + L_w + L_i + 3} - \sum_{i=1}^n r_i \sigma_i (\sqrt{L_w + L_i + 2} - \sqrt{L_i + 1}) y_{1i} \\ &\quad + \sum_{i=1}^n r_i \sigma_i (\sqrt{L_s + L_w + L_i + 3} - \sqrt{L_w + L_i + 2})(y_{1i} + y_{2i}). \end{aligned} \tag{10}$$

We then denote $r_s^2(L_s + 1) = b_2$, $r_w^2(L_w + 1) = b_1$, $\sigma_i^2 = a_i$, $\sum_{i=1}^n r_i \sigma_i \sqrt{(L_s + L_w + L_i + 3)} = C$, $r_i \sigma_i (\sqrt{(L_w + L_i + 2)} - \sqrt{(L_i + 1)}) = c_{1i}$, $r_i \sigma_i (\sqrt{(L_s + L_w + L_i + 3)} - \sqrt{(L_w + L_i + 2)}) = c_{2i}$. Let $y_{1i} + y_{2i} = z_{2i}$, $y_{1i} = z_{1i}$, and then we have

$$SC_3(z) = SC_2(y) = \sum_{j=1}^2 \left(b_j \sum_{i=1}^n a_i z_{ji} \right)^{1/2} - \sum_{j=1}^2 \sum_{i=1}^n c_{ji} z_{ji} + C, \tag{11}$$

where $z = (z_{11}, z_{12}, \ldots, z_{1n}, z_{21}, \ldots, z_{2n}) \in \{0, 1\}^{2n}$ and $z_{2i} \geqslant z_{1i}$, for any $i = 1, 2, \ldots, n$. Since C is a constant, minimizing $SC_3(z)$ in Equation (11) is clearly a special case of Problem (1) with $s = 1/2$.

4 Extension

In this section, we consider an extended version of Problem (1):

$$\text{Min} \quad f(x) = \sum_{j=1}^{m} \left(b_j \sum_{i=1}^{n} a_i x_{ji} \right)^s - \sum_{j=1}^{m} \sum_{i=1}^{n} c_{ji} x_{ji}, \tag{12}$$

$$\text{s.t.} \quad x_{ki} \geqslant x_{li}, \quad \text{for any } k > l, i = 1, 2, \ldots, n, \quad x_{ji} \in \{0, 1\}, \quad \text{for any } i \text{ and } j,$$

where $0 < s < 1$, $a_i > 0$, $b_j > 0$, $c_{ji} > 0$, and n and m are both integers. Here we assume that n is much larger than m, and in the rest of the paper, we want to find an algorithm to solve Problem (12) in a polynomial time of n.

In this section, we refer to x as a feasible solution if x satisfies constraints of Problem (12). For any feasible solution x, define

$$\begin{aligned}
&S_1(x) = \{i \mid x_{1i} = x_{2i} = \cdots = x_{mi} = 1\},\\
&S_k(x) = \{i \mid x_{1i} = \cdots = x_{k-1,i} = 0, x_{ki} = \cdots = x_{mi} = 1\}, \quad k = 2, 3, \ldots, m,\\
&S_{m+1}(x) = \{i \mid x_{1i} = x_{2i} = \cdots = x_{mi} = 0\}.
\end{aligned}$$

Let $T_0(x) = \emptyset$, $T_k(x) = \bigcup_{l=1}^{k} S_l(x)$, $k = 1, 2, \ldots, m+1$. Clearly, we have $T_k(x) = \{i \mid x_{ki} = 1\}$ $(k = 1, 2, \ldots, m)$ and $T_{m+1}(x) = S = \{1, 2, \ldots, n\}$. Thus, Equation (12) is equivalent to

$$f(x) = \sum_{j=1}^{m} \left[\left(b_j \sum_{i \in T_j(x)} a_i \right)^s - \sum_{i \in T_j(x)} c_{ji} \right] = \sum_{j=1}^{m} \left[\left(\sum_{i \in T_{j+1}(x) \backslash S_{j+1}(x)} a_i b_j \right)^s - \sum_{i \in T_{j+1}(x) \backslash S_{j+1}(x)} c_{ji} \right]. \tag{13}$$

Note that a feasible solution of Problem (12) can uniquely determine a set partition of S, i.e., $\{P_1, \ldots, P_{m+1}\}$ with properties $\bigcup_{k=1}^{m+1} P_k = S$ and $P_k \cap P_l = \emptyset$ for any $k \neq l$. Conversely, a set partition of S, $\{P_1, \ldots, P_{m+1}\}$, can also uniquely determine a feasible solution of Problem (12). Therefore, to determine the optimal solution of Problem (12), one only needs to choose $\{P_1, \ldots, P_{m+1}\}$ to minimize

$$\tilde{f}(P_1, \ldots, P_{m+1}) = \sum_{j=1}^{m} \left[\left(b_j \sum_{i \in G_{j+1} \backslash P_{j+1}} a_i \right)^s - \sum_{i \in G_{j+1} \backslash P_{j+1}} c_{ji} \right], \tag{14}$$

$$\text{s.t.} \quad G_j = G_{j+1} \backslash P_{j+1}, j = 1, 2, \ldots, m, G_{m+1} = S, \bigcup_{k=1}^{m+1} P_k = S.$$

Therefore, Problem (12) can be solved by a dynamic programming method. Denote

$$\begin{aligned}
V_k(G_k) &= \min_{P_2, \ldots, P_k} \sum_{j=2}^{k} \left[\left(b_j \sum_{i \in G_j \backslash P_j} a_i \right)^s - \sum_{i \in G_j \backslash P_j} c_{ji} \right], \quad k = 2, 3, \ldots, m+1,\\
v_k(G_k, P_k) &= \left(b_j \sum_{i \in G_k \backslash P_k} a_i \right)^s - \sum_{i \in G_k \backslash P_k} c_{ji}, \quad k = 2, 3, \ldots, m+1.
\end{aligned}$$

Then the Bellman equation for this problem is

$$\begin{cases}
V_k(G_k) = \min\limits_{P_k \subseteq G_k} \{v_k(P_k, G_k) + V_{k-1}(G_{k-1})\}, \quad k = 2, \ldots, m+1,\\
V_1(G_1) = 0, \quad \text{for any } G_1 \subseteq S,
\end{cases} \tag{15}$$

where the decision variables are P_k, $k = 2, \ldots, m+1$, and the state transformation functions are

$$G_{k-1} = G_k \backslash P_k, \quad k = 2, \ldots, m+1. \tag{16}$$

Proposition 4.1. *Let $\{P_2^*, \ldots, P_{m+1}^*\}$ be the solution of the Bellman equation (15) with $G_{m+1} = S$ and $V_{m+1}^*(S)$ be the corresponding value when $P_2 = P_2^*, \ldots, P_{m+1} = P_{m+1}^*$. Denote $P_1^* = S \backslash (\bigcup_{k=2}^{m+1} P_k^*)$. Then the optimal value of Problem (12) is $V_{m+1}^*(S)$ and the optimal solution for Problem (12) is $x^* =$*

$(x^*_{11}, x^*_{12}, \ldots, x^*_{1n}, \ldots, x^*_{m1}, x^*_{m2}, \ldots, x^*_{mn})$, *where* $x^*_{1i} = x^*_{2i} = \cdots = x^*_{mi} = 1$ *for* $i \in P^*_1$; $x^*_{1i} = x^*_{2i} = \cdots = x^*_{k-1,i} = 0$, $x^*_{ki} = \cdots = x^*_{mi} = 1$, *for* $i \in P^*_k$ $(k = 1, 2, \ldots, m)$; $x^*_{1i} = x^*_{2i} = \cdots = x^*_{mi} = 0$ *for* $i \in P^*_{m+1}$.

Proof. It is obvious from the analysis above. □

Generally, the Bellman equation (15) is not polynomial-time solvable. However, the following theorem shows that, to find the solution of the Bellman equation (15), we only need to confine our searchings to the feasible solutions with special structure. This will considerably decrease the computational complexity.

Define $R_{k,l}(i)$, for any $1 \leqslant k \leqslant l \leqslant m$, as the ranking position of the item i in the ascending order by $(\sum_{j=k}^{l} c_{ji})/a_i$. By comparison with $R_1(i)$, $R_2(i)$, $R_3(i)$ defined in Section 2, we have that $R_{1,1}(i)$ here is equivalent to $R_1(i)$, $R_{2,2}(i)$ is equivalent to $R_2(i)$, $R_{1,2}(i)$ is equivalent to $R_3(i)$.

Theorem 4.1. *Assume* $R_{v,w}(i)$ $(1 \leqslant v \leqslant w \leqslant m, i = 1, 2, \ldots, n)$ *is determined. Any optimal solution* x^* *for Problem* (12) *satisfies that*: *For any* $1 \leqslant v \leqslant w \leqslant m$, $p \in S_{w+1}(x^*)$, *and* $q \in S_v(x^*)$, *we have* $R_{v,w}(p) < R_{v,w}(q)$.

Proof. The proof of this theorem is similar to that of Theorem 2.1. □

By Theorem 4.1 and Proposition 4.1, we have that the solution of the Bellman equation (15), $\{P^*_2, \ldots, P^*_{m+1}\}$, together with $P^*_1 = S \backslash (\bigcup_{k=2}^{m+1} P^*_k)$, possesses the following property: For any $1 \leqslant v \leqslant w \leqslant m$, $p \in P^*_{w+1}$, and $q \in P^*_v$, we have $R_{v,w}(p) < R_{v,w}(q)$.

Suppose that $\{P^*_2, \ldots, P^*_{m+1}\}$ is the solution of the Bellman equation (15) and $P^*_1 = S \backslash (\bigcup_{k=2}^{m+1} P^*_k)$. Let $k^*_{1,m} = \max\{R_{1,m}(i) \mid i \in P^*_{m+1}\}$ and $Q_{1,m} = \{i \in S \mid R_{1,m}(i) \leqslant k^*_{1,m}\}$. Then we have $P^*_{m+1} \subseteq Q_{1,m}$ and that $Q_{1,m} \cap P^*_1 = \emptyset$ (This statement can be proved based on Theorem 4.1 with a similar argument to that of Part (a) of Corollary 2.1).

Let $k^*_{2,m} = \max\{R_{2,m}(i) \mid i \in P^*_{m+1}\}$ and $Q_{2,m} = \{i \in Q_{1,m} \mid R_{2,m}(i) \leqslant k^*_{2,m}\}$. Similarly to Corollary 2.1, we have $P^*_{m+1} \subseteq Q_{2,m}$ and $Q_{2,m} \cap P^*_2 = \emptyset$.

Continuing this process, we can also define $Q_{k,m}$, with similar properties, $P^*_{m+1} \subseteq Q_{k,m}$ and $Q_{k,m} \cap P^*_k = \emptyset$, $k = 1, 2, \ldots, m$, as above. Note that $P^*_{m+1} \subseteq Q_{m,m}$ and $Q_{m,m} \cap (\bigcup_{k=1}^{m} P^*_k) = \emptyset$. Therefore, $P^*_{m+1} = Q_{m,m}$.

Next, consider $G_m = S \backslash P^*_{m+1}$ (correspondingly, consider $R_{k,m-1}(i)$, $k = 1, 2, \ldots, m-1$, instead of $R_{k,m}(i)$, $k = 1, 2, \ldots, m$). By a similar method, we can define $Q_{m-1,m-1} = P^*_m$. Proceeding similarly, we can define $Q_{m-2,m-2} = P^*_{m-1}, \ldots, Q_{1,1} = P^*_2$ (P^*_1 can be defined by $P^*_1 = S \backslash (\bigcup_{k=2}^{m+1} P^*_k)$).

By the analysis above, we are able to give the algorithm to solve the Bellman equation (15) and Problem (12). Before presenting this algorithm, we descriptively provide a function, which will be called in this algorithm. This function runs based on given $R_{v,w}(i)$, $1 \leqslant v \leqslant w \leqslant m$, $i = 1, 2, \ldots, n$.

Function. $[P_1, \ldots, P_k, V_k] = \bar{f}(k, G_k)$.
If $k = 1$, then let $P_1 = G_1$ and $V_1 = 0$.
If $k > 1$, then proceed as follows.
Let $P_1 = \cdots = P_{k-1} = \emptyset$, $P_k = G_k$, $V_k = 0$.
For $p_1 = 1 : |G_k|$
  Choose $Q_{1,k-1} = \{i_1, i_2, \ldots, i_{p_1}\} \subseteq G_k$ such that $R_{1,k-1}(i_1), R_{1,k-1}(i_2), \ldots, R_{1,k-1}(i_{p_1})$ are the p_1 smallest numbers in the set $\{R_{1,k-1}(i) \mid i \in G_k\}$.
  For $p_2 = 1 : p_1$
    Choose $Q_{2,k-1} = \{i_1, i_2, \ldots, i_{p_2}\} \subseteq Q_{1,k-1}$ such that $R_{2,k-1}(i_1)$, $R_{2,k-1}(i_2), \ldots, R_{2,k-1}(i_{p_2})$ are the p_2 smallest numbers in the set $\{R_{2,k-1}(i) \mid i \in Q_{1,k-1}\}$.
    $\vdots$
    For $p_{k-1} = 1 : p_{k-2}$
      Choose $Q_{k-1,k-1} = \{i_1, i_2, \ldots, i_{p_{k-1}}\} \subseteq Q_{k-2,k-1}$ such that $R_{k-1,k-1}(i_1), R_{k-1,k-1}(i_2), \ldots,$ $R_{k-1,k-1}(i_{p_{k-1}})$ are the p_{k-1} smallest numbers in the set $\{R_{k-1,k-1}(i) \mid i \in Q_{k-2,k-1}\}$.
      Let $G_{k-1} = G_k \backslash Q_{k-1,k-1}$.
      Call Function $\bar{f}$ and let $[\bar{P}_1, \ldots, \bar{P}_{k-1}, \bar{V}_{k-1}] = \bar{f}(k-1, G_{k-1})$.
      If $\bar{V}_{k-1} + (\sum_{i \in G_k \backslash Q_{k-1,k-1}} a_i b_k)^s - \sum_{i \in G_k \backslash Q_{k-1,k-1}} c_{ki} < V_k$, then let $P_1 = \bar{P}_1, \ldots, P_{k-1} = \bar{P}_{k-1}$, $P_k = Q_{k-1,k-1}$ and $V_k = \bar{V}_{k-1} + (\sum_{i \in G_k \backslash Q_{k-1,k-1}} a_i b_k)^s - \sum_{i \in G_k \backslash Q_{k-1,k-1}} c_{ki}$.

End
⋮
End
End

With the function above, we have the following algorithm.

Algorithm 4.1.

Step 1: Solve $R_{v,w}(i)$, $1 \leqslant v \leqslant w \leqslant m$, $i = 1, 2, \ldots, n$, by some sorting procedure.
Let $P_1^* = \cdots = P_m^* = \emptyset$, $P_{m+1}^* = S$, $V_{m+1}^* = 0$.

Step 2: Call Function $[P_1^*, \ldots, P_{m+1}^*, V_{m+1}^*] = \bar{f}(m+1, S)$.

Step 3: Define $x^* = (x_{11}^*, \ldots, x_{1n}^*, \ldots, x_{m1}^*, \ldots, x_{mn}^*)$, where $x_{1i}^* = \cdots = x_{mi}^* = 1$ for $i \in P_1^*$; $x_{1i}^* = \cdots = x_{k-1,i}^* = 0$, $x_{ki}^* = \cdots = x_{mi}^* = 1$, for $i \in P_k^*$ $(k = 1, 2, \ldots, m)$; $x_{1i}^* = \cdots = x_{m,i}^* = 0$ for $i \in P_{m+1}^*$.
Output x^* as the optimal solution and V_{m+1}^* as the optimal value for Problem (12).

Theorem 4.2. *Algorithm* 4.1 *is able to find the optimal solution of Problem* (12) *with a complexity of* $O(n^{(m^2+m)/2})$.

Proof. By Theorem 4.1 and the associated analysis presented previously, one can easily show that the optimal solution of Problem (12) is examined in Algorithm 4.1. Therefore, Algorithm 4.1 is able to find the optimal solution for Problem (12).

Step 1 of Algorithm 4.1 runs with a complexity of $O(m^2 n \log n)$. In Step 2, assume the function $\bar{f}(k, G_k)$ runs with $g(k)$ calculations. Thus, by analysis of Function $\bar{f}$, we know that, for any given $k \in \{1, 2, \ldots, m\}$, $\bar{f}(k+1, G_{k+1})$ runs with $g(k+1) = p_{k-1} \cdot p_{k-2} \ldots p_1 \cdot |G_k| g(k)$ calculations. Since $p_{k-1} \leqslant p_{k-2} \leqslant \cdots \leqslant p_1 \leqslant |G_k| \leqslant n$, we have $g(k+1) \leqslant n^k g(k)$. Therefore, $g(m+1) = O(n^m \cdot n^{m-1} \cdot \cdots \cdot n) = O(n^{(m^2+m)/2})$, i.e., Algorithm 4.1 runs with a complexity of $O(n^{(m^2+m)/2})$. □

With the method of exhaustion, the running time for solving Problem (12) would be increasing exponentially with n, which means that the method is inefficient. Theorem 4.2 indicates that Algorithm 4.1 is an effective method for solving Problem (12) compared with the method of exhaustion, especially when n is large. It can be shown that similar problems as in Section 3 for an m-level supply chain ($m \geqslant 3$) can also be reduced to a special case of Problem (12). Thus, Algorithm 4.1 can also be applied to solving the optimal solution for an m-level supply chain with early order commitment.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 70471008, 70971072).

References

1 Bilitzky A, Sadeh A. Efficient solutions for special zero-one programming problems. J Combin Optim, 2005, 10: 227–238
2 Fao A, Mauricio G C, Resende A. A probabilistic heuristic for a computationally difficult set covering problem. Oper Res Lett, 1989, 8: 67–71
3 Huang W C, Kao C Y, Hong J T. A genetic algorithm for set covering problems. IEEE Int Conf Genet Algor, 1994: 569–574
4 Nemhauser G L, Wolsey L A. Interger and Combinatorial Optimization. New York: John Wiley and Sons, 1988
5 Padberg M W. Perfect zero-one matrices. Math Program, 1974, 6: 180–196
6 Padberg M W. Lehman's forbidden minor characterization of ideal 0-1 matrices. Discrete Math, 1993, 111: 409–420
7 Wolsey L A. Integer Programming. Intersince Series in Discrete Mathematics and Optimization. New York: John Wiley and Sons, 1998
8 Xie J, Zhou D, Wei J C, et al. Price discount based on early order commitment in a single manufacturer-multiple retailer supply chain. European J Oper Res, 2010, 200: 368–376
9 Xiong H, Xie J, Wang M. A note on "Price discount based on early order commitment in a single-manufacturer-multiple-retailer supply chain". European J Oper Res, to appear
10 Zhao X, Xie J, Wei J C. The impact of forecast errors on early order commitment in a supply chain. Decis Sci, 2002, 33: 251–280
11 Zhao X, Xie J, Wei J C. The value of early order commitment in a two-level supply chain. European J Oper Res, 2007, 180: 194–214